

Τμήμα Διοίκησης Επιχειρήσεων και Οργανισμών

## Εργασία Μαθήματος: “Σχεδιασμός και Ανάπτυξη Διαδικτυακών Εφαρμογών”

### Θέμα Εργασίας: **e-shop Meli Evangelikon** Μελισσοκομικά (Προϊόντα Κυψέλης)



Όνομα Ομάδας: **MeliEvangelikon A.E.**

**Μέλη Ομάδας:**

Ευαγγελία Χανιωτάκη 1344202000108

Αντώνης Παναγιώτης Βερβαινιώτης 1344202000007

**Αθήνα, 2023**

## 1. Θέμα και πηγές συλλογής απαιτήσεων

Το θέμα το οποίο έχει επιλέξει η ομάδα μας είναι η κατασκευή ενός e-shop με τα προϊόντα που παράγει η οικογένεια Παππά . Το όνομα της επιχείρησης είναι Μέλι Ευαγγελικών . Διαλέξαμε αυτό το θέμα μιας και το ένα μέλος της ομάδας μας είναι ο παραγωγός αυτών των προϊόντων και έχει την αρίστη κατάρτιση . Σκοπός της επιλογής αυτής ήταν να μπορέσουμε να αφιερώσουμε αυτόν τον χρόνο που εξοικονομήθηκε για την καλύτερη τεχνική κατάρτιση μας. Παρόλα αυτά μελετήθηκαν στο διαδίκτυο και άλλα websites με παρόμοια προϊόντα ώστε να αποκτήσουμε μια ειδικότερη εικόνα για την προβολή των προϊόντων αλλά και τον τρόπο προώθησης τους .

## 2. Σχεδιασμός υψηλού επιπέδου

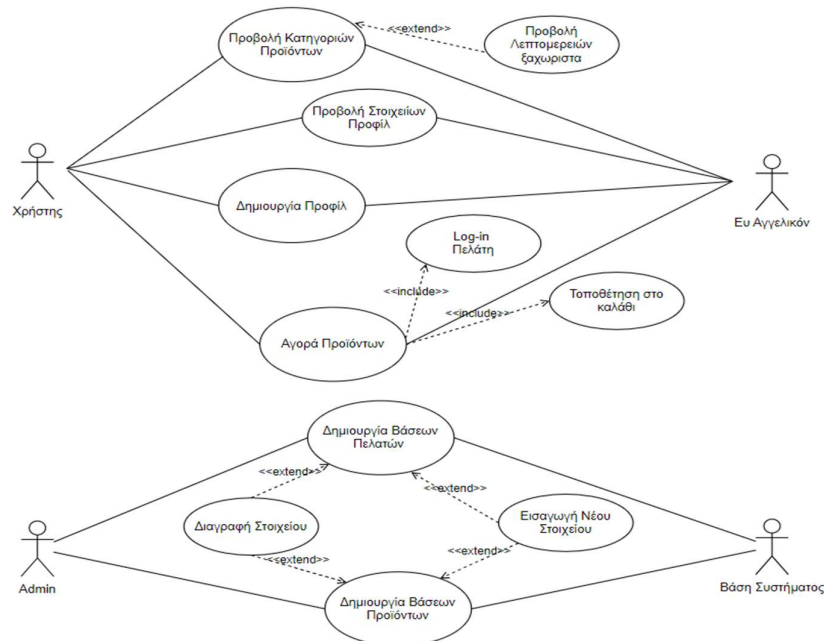
Ο χρήστης που μπαίνει στην ιστοσελίδα μπορεί να δημιουργήσει προφίλ, έναν λογαριασμό στην ιστοσελίδα και να μπορεί να βλέπει τα στοιχεία του, πχ την συνδρομή του.

Ο χρήστης μπορεί να δει τα προϊόντα ονομαστικά και αν επιθυμεί να μάθει λεπτομέρειες για κάποιο συγκεκριμένο.

Η αγορά προϋποθέτει να είναι συνδεδεμένος ο πελάτης στον λογαριασμό του, να τοποθετήσει τα προϊόντα που επιθυμεί στο καλάθι και να επιλέξει τον τρόπο πληρωμής για να ολοκληρωθεί η παραγγελία.

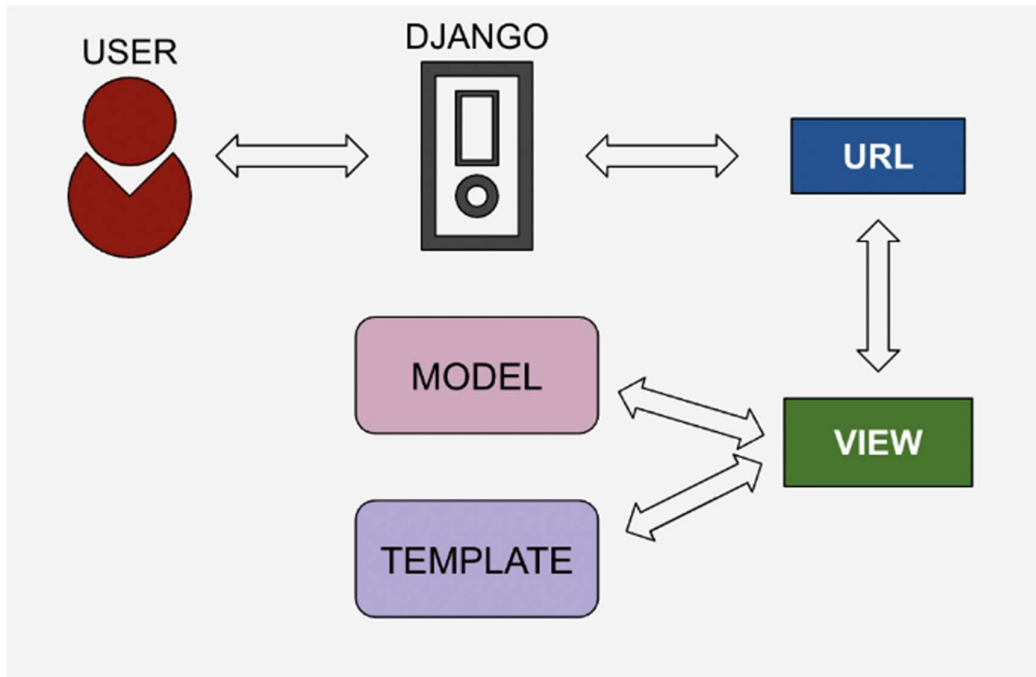
Ο Admin δημιουργεί μια βάση και για τα προϊόντα και για τους χρήστες από την οποία μπορεί να εισάγει, να διαγράψει ή να επεξεργαστεί ένα ή περισσότερα στοιχεία.

Επιπλέον, μπορεί να διαχειρίζεται την παραγγελία που κάνει ένας χρήστης.



Διάγραμμα Use Case

### 3. Αρχιτεκτονικός Σχεδιασμός



Το MVT είναι μια αρχιτεκτονική που χρησιμοποιείται στο Django για τον οργανισμό και τον χειρισμό των εφαρμογών web. Ο τρόπος λειτουργίας του MVT του Django ορίζετε από:

- **Μοντέλα (Models):** Τα μοντέλα αντιπροσωπεύουν τα δεδομένα της εφαρμογής σας. Τα μοντέλα καθορίζουν τη δομή της βάσης δεδομένων, τα πεδία και τις σχέσεις μεταξύ των διάφορων πινάκων.
- **Προβολές (Views):** Οι προβολές είναι υπεύθυνες για την αλληλεπίδραση με τα αιτήματα του χρήστη και την προετοιμασία των δεδομένων για προβολή. Ορίζει τις προβολές ως συναρτήσεις ή κλάσεις Python που λαμβάνουν ένα αίτημα HTTP και επιστρέφουν μια απόκριση. Οι προβολές μπορούν να επικοινωνούν με τα μοντέλα για να ανακτήσουν ή να αποθηκεύσουν δεδομένα και να επιλέξουν το πρότυπο που θα προβληθεί.
- **Πρότυπα (Templates):** Τα πρότυπα καθορίζουν την εμφάνιση των δεδομένων που προβάλλονται στο χρήστη. Ορίζετε πρότυπα ως αρχεία HTML με ενσωματωμένες ετικέτες Django. Οι ετικέτες Django επιτρέπουν την παρουσίαση δυναμικών δεδομένων, την επανάληψη, την απόφαση και άλλων λειτουργιών.

Κατά την επεξεργασία μιας αίτησης, το Django παρέχει τον μηχανισμό δρομολόγησης (routing) για να αντιστοιχίσει το URL αίτησης σε μια συγκεκριμένη προβολή. Η προβολή επεξεργάζεται το αίτημα, ανακτά δεδομένα από τα μοντέλα, αποφασίζει ποιο πρότυπο θα χρησιμοποιηθεί και επιστρέφει μια απόκριση στο χρήστη.

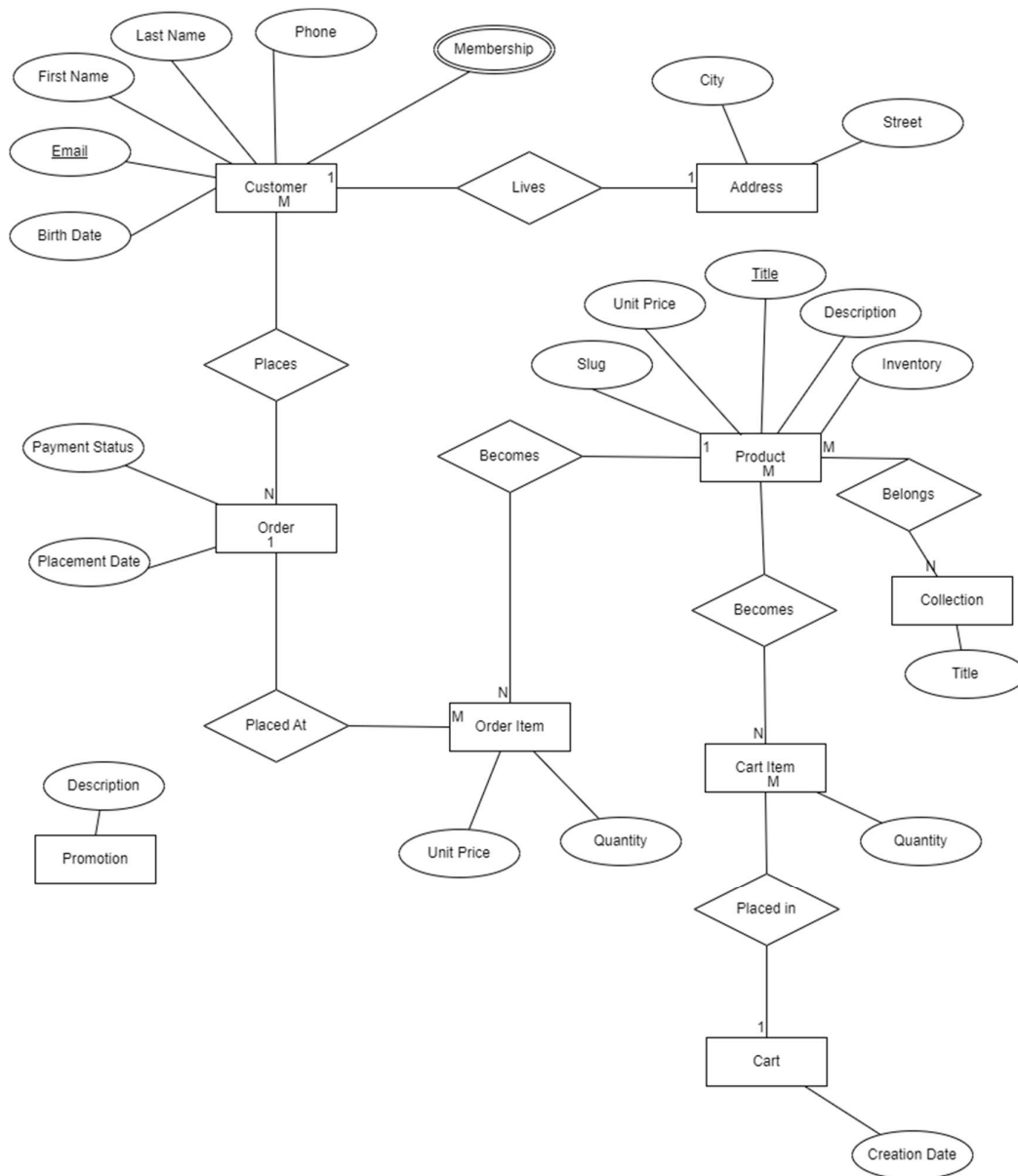
Το MVT του Django επιτρέπει την οργανωμένη και αποτελεσματική ανάπτυξη εφαρμογών web.

#### 4.Οντότητες, συσχετίσεις, παραδοχές και ER

Ένας πελάτης με πρωτεύον κλειδί το email του μένει σε μια διεύθυνση. Πολλοί πελάτες μπορούν να κάνουν πολλές παραγγελίες.

Το προϊόν με πρωτεύον κλειδί το όνομα του ανήκει σε μια συλλογή. Πολλά προϊόντα μπορούν να μπαίνουν στην κάρτα και να γίνονται αντικείμενα της παραγγελίας.

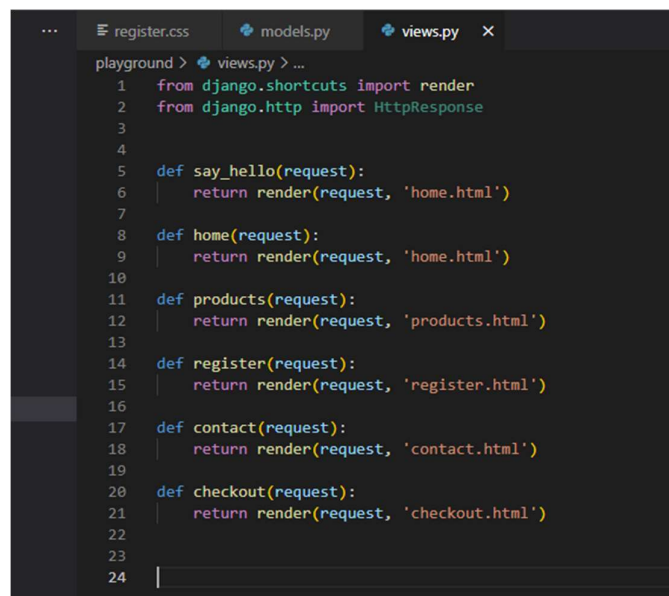
Τα αντικείμενα της κάρτας μπαίνουν σε μία κάρτα.



## 5. Δημιουργία views

Ο κώδικας μας μέσα στο αρχείο `views.py` όπου βρίσκετε στη κεντρική μας εφαρμογή (`playground`) παρουσιάζεται η χρήση και η λειτουργία κάθε μεθόδου προβολής:

- `say_hello(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"home.html"`. και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"home.html"`.
- `home(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"home.html"`. και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"home.html"`.
- `products(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"products.html"` και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"products.html"`.
- `register(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"register.html"` και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"register.html"`.
- `contact(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"contact.html"` και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"contact.html"`.
- `checkout(request)`: Χρησιμοποιείται για την προβολή της σελίδας `"checkout.html"` και επιστρέφει μια απόκριση (`HttpResponse`) που αναφέρεται στο περιεχόμενο της σελίδας `"checkout.html"`.



```
... register.css models.py views.py X
playground > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4
5  def say_hello(request):
6      return render(request, 'home.html')
7
8  def home(request):
9      return render(request, 'home.html')
10
11 def products(request):
12     return render(request, 'products.html')
13
14 def register(request):
15     return render(request, 'register.html')
16
17 def contact(request):
18     return render(request, 'contact.html')
19
20 def checkout(request):
21     return render(request, 'checkout.html')
22
23
24
```

Στιγμιότυπο `playground>views.py`

Οι μέθοδοι προβολής είναι υπεύθυνες για την επεξεργασία των αιτημάτων HTTP και την προβολή των αντίστοιχων σελίδων HTML στον περιηγητή του χρήστη. Αυτό επιτυγχάνεται μέσω της συνάρτησης `render()`, η οποία δέχεται το αίτημα ως παράμετρο και τον ορίζει ως πρώτο όρισμα σε συνδυασμό με το όνομα του αρχείου HTML που πρέπει να προβληθεί. Έπειτα, επιστρέφεται μια απόκριση (`HttpResponse`) με το περιεχόμενο της σελίδας HTML που πρέπει να προβληθεί.

## 6.Δημιουργία πινάκων

Μέσα στο αρχείο μας `models.py` όπου βρίσκετε στην εφαρμογή (`store`) , έχουμε δημιουργήσει 9 πίνακες-κλάσεις. Οι κλάσεις-πίνακες είναι `Promotion` , `Collection` , `Product` , `Customer` , `Order` , `Order Item` , `Address` , `Cart` και `Cart Item`. Το όνομα κάθε πίνακα είναι πλήρως περιγραφικό αυτού του οποίου περιέχει.

### Promotion

Η κλάση "**Promotion**" αναπαριστά μια προώθηση ή μια ειδική προσφορά και περιλαμβάνει πληροφορίες όπως η περιγραφή της προώθησης και το ποσοστό έκπτωσης που προσφέρει.

- 1) **description**: Αλφαριθμητικό πεδίο (`CharField`) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για την περιγραφή της προώθησης, δηλαδή για να περιγράψει την ειδική προσφορά ή έκπτωση που σχετίζεται με την προαγωγή.
- 2) **discount**: Αριθμητικό πεδίο (`FloatField`) που αντιπροσωπεύει το ποσοστό έκπτωσης της προώθησης. Χρησιμοποιείται για να αποθηκεύσει το ποσοστό μείωσης τιμής που εφαρμόζεται στο προϊόν κατά τη διάρκεια της προαγωγής.

### Collection

Η κλάση "**Collection**" αναπαριστά μια συλλογή προϊόντων και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **title**: Αλφαριθμητικό πεδίο (`CharField`) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για τον τίτλο της συλλογής.
- 2) **featured\_product**: Εξωτερικό κλειδί (`ForeignKey`) που συνδέει τη συλλογή με ένα προϊόν (`Product`). Η επιλογή `on_delete=models.SET_NULL` σημαίνει ότι αν το προϊόν διαγραφεί, το γνώρισμα `featured_product` θα οριστεί σε `null` (άδειο). Η επιλογή `null=True` επιτρέπει την αποθήκευση της τιμής `null` για το γνώρισμα `featured_product`. Το `related_name='+'` αποτρέπει τη δημιουργία αντικειμένου αντίστροφης σχέσης από το προϊόν προς τη συλλογή.

### Product

Η κλάση "**Product**" αναπαριστά ένα προϊόν στη βάση δεδομένων και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **title**: Αλφαριθμητικό πεδίο (`CharField`) με μέγιστο μήκος 255 χαρακτήρων , χρησιμοποιείται για τον τίτλο του προϊόντος.
- 2) **slug**: Αλφαριθμητικό πεδίο (`SlugField`) που χρησιμοποιείται για τη μοναδική αναγνωριστική συμβολοσειρά του προϊόντος.
- 3) **description**: Πεδίο κειμένου (`TextField`) που επιτρέπει την αποθήκευση μιας περιγραφής του προϊόντος. Μπορεί να είναι `null` (κενό) ή `blank` (άδειο).
- 4) **unit\_price**: Αριθμητικό πεδίο (`DecimalField`) που αντιπροσωπεύει την τιμή μονάδας του προϊόντος. Έχει μέγιστο πλήθος ψηφίων 6 και 2 δεκαδικά ψηφία. Χρησιμοποιείται επίσης ένας ελάχιστος επικερδής ελέγχος (`validators`) για να διασφαλιστεί ότι η τιμή δεν μπορεί να είναι μικρότερη από 1.

- 5) **inventory**: Ακέραιο πεδίο (IntegerField) που αντιπροσωπεύει το απόθεμα του προϊόντος. Έχει έναν ελάχιστο επικερδή έλεγχο για να διασφαλιστεί ότι η τιμή δεν μπορεί να είναι αρνητική.
- 6) **last\_update**: Ημερομηνία και ώρα (DateTimeField) που αυτόματα ενημερώνεται με την τρέχουσα ώρα και ημερομηνία κάθε φορά που το αντικείμενο του προϊόντος αποθηκεύεται.
- 7) **collection**: Εξωτερικό κλειδί (ForeignKey) που συνδέει το προϊόν με μια συλλογή (Collection). Η επιλογή `on_delete=models.PROTECT` σημαίνει ότι αν η συλλογή διαγραφεί, τα προϊόντα που την αναφέρουν θα προστατευθούν και η διαγραφή δεν θα επιτραπεί.
- 8) **promotions**: Πεδίο πολλαπλής επιλογής (ManyToManyField) που συνδέει το προϊόν με πολλές προωθητικές ενέργειες (Promotion). Η επιλογή `blank=True` σημαίνει ότι δεν είναι υποχρεωτικό να έχει επιλεγεί κάποια προωθητική ενέργεια για το προϊόν.

## Customer

Η κλάση "Customer" αναπαριστά έναν πελάτη και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **first\_name**: Αλφαριθμητικό πεδίο (CharField) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για το όνομα του πελάτη.
- 2) **last\_name**: Αλφαριθμητικό πεδίο (CharField) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για το επίθετο του πελάτη.
- 3) **email**: Πεδίο email (EmailField) που αντιπροσωπεύει το email του πελάτη. Έχει τη μοναδική ιδιότητα (`unique=True`), οπότε κάθε πελάτης πρέπει να έχει μοναδικό email στο σύστημα.
- 4) **phone**: Αλφαριθμητικό πεδίο (CharField) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για τον αριθμό τηλεφώνου του πελάτη.
- 5) **birth\_date**: Πεδίο ημερομηνίας (DateField) που αντιπροσωπεύει την ημερομηνία γέννησης του πελάτη. Μπορεί να είναι null (κενό) ή blank (άδειο).
- 6) **membership**: Αλφαριθμητικό πεδίο (CharField) με μήκος 1 χαρακτήρα. Χρησιμοποιείται για να αναπαραστήσει την επίπεδη μέληση του πελάτη. Έχει περιορισμένες επιλογές μέλους (MEMBERSHIP\_CHOICES) που έχουν οριστεί ως σταθερές (MEMBERSHIP\_BRONZE, MEMBERSHIP\_SILVER, MEMBERSHIP\_GOLD). Το πεδίο έχει προεπιλεγμένη τιμή MEMBERSHIP\_BRONZE.

## Order

Η κλάση "Order" αναπαριστά μια παραγγελία και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **placed\_at**: Ημερομηνία και ώρα (DateTimeField) που αυτόματα ορίζεται στην τρέχουσα ημερομηνία και ώρα κάθε φορά που δημιουργείται μια παραγγελία.
- 2) **payment\_status**: Αλφαριθμητικό πεδίο (CharField) με μήκος 1 χαρακτήρα. Χρησιμοποιείται για να αναπαραστήσει την κατάσταση πληρωμής της παραγγελίας. Έχει περιορισμένες επιλογές κατάστασης πληρωμής (PAYMENT\_STATUS\_CHOICES) που έχουν οριστεί ως σταθερές (PAYMENT\_STATUS\_PENDING, PAYMENT\_STATUS\_COMPLETE,

PAYMENT\_STATUS\_FAILED). Το πεδίο έχει προεπιλεγμένη τιμή PAYMENT\_STATUS\_PENDING.

- 3) **customer:** Εξωτερικό κλειδί (ForeignKey) που συνδέει την παραγγελία με έναν πελάτη (Customer). Η επιλογή `on_delete=models.PROTECT` σημαίνει ότι αν ο πελάτης διαγραφεί, η παραγγελία θα προστατευθεί και η διαγραφή δεν θα επιτραπεί

### Order Item

Η κλάση "Order Item" αναπαριστά ένα αντικείμενο μίας παραγγελίας και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **order:** Εξωτερικό κλειδί (ForeignKey) που συνδέει το αντικείμενο με μια παραγγελία (Order).
- 2) **product:** Εξωτερικό κλειδί (ForeignKey) που συνδέει το αντικείμενο με ένα προϊόν (Product).
- 3) **quantity:** Ακέραιο πεδίο (PositiveIntegerField) που αντιπροσωπεύει την ποσότητα του προϊόντος που παραγγέλθηκε.
- 4) **unit\_price:** Αριθμητικό πεδίο (DecimalField) που αντιπροσωπεύει την τιμή μονάδας του προϊόντος.

### Address

Η κλάση "Address" αναπαριστά την διεύθυνση και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 1) **street:** Αλφαριθμητικό πεδίο (CharField) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για την οδό της διεύθυνσης.
- 2) **city:** Αλφαριθμητικό πεδίο (CharField) με μέγιστο μήκος 255 χαρακτήρων. Χρησιμοποιείται για την πόλη της διεύθυνσης.
- 3) **customer:** Εξωτερικό κλειδί (ForeignKey) που συνδέει το αντικείμενο με έναν πελάτη (Customer)

### Cart

Η κλάση "Cart" αναπαριστά το καλάθι και περιλαμβάνει το παρακάτω γνώρισμα:

- 1) **created\_at:** Ημερομηνία και ώρα (DateTimeField) που αυτόματα ορίζεται στην τρέχουσα ημερομηνία και ώρα κάθε φορά που δημιουργείται ένα καλάθι (Cart).

### Cart Item

Η κλάση "Cart Item" αναπαριστά προϊόν του καλαθιού και περιλαμβάνει τα παρακάτω γνωρίσματα:

- 2) **cart:** Εξωτερικό κλειδί (ForeignKey) που συνδέει το αντικείμενο με ένα καλάθι (Cart).



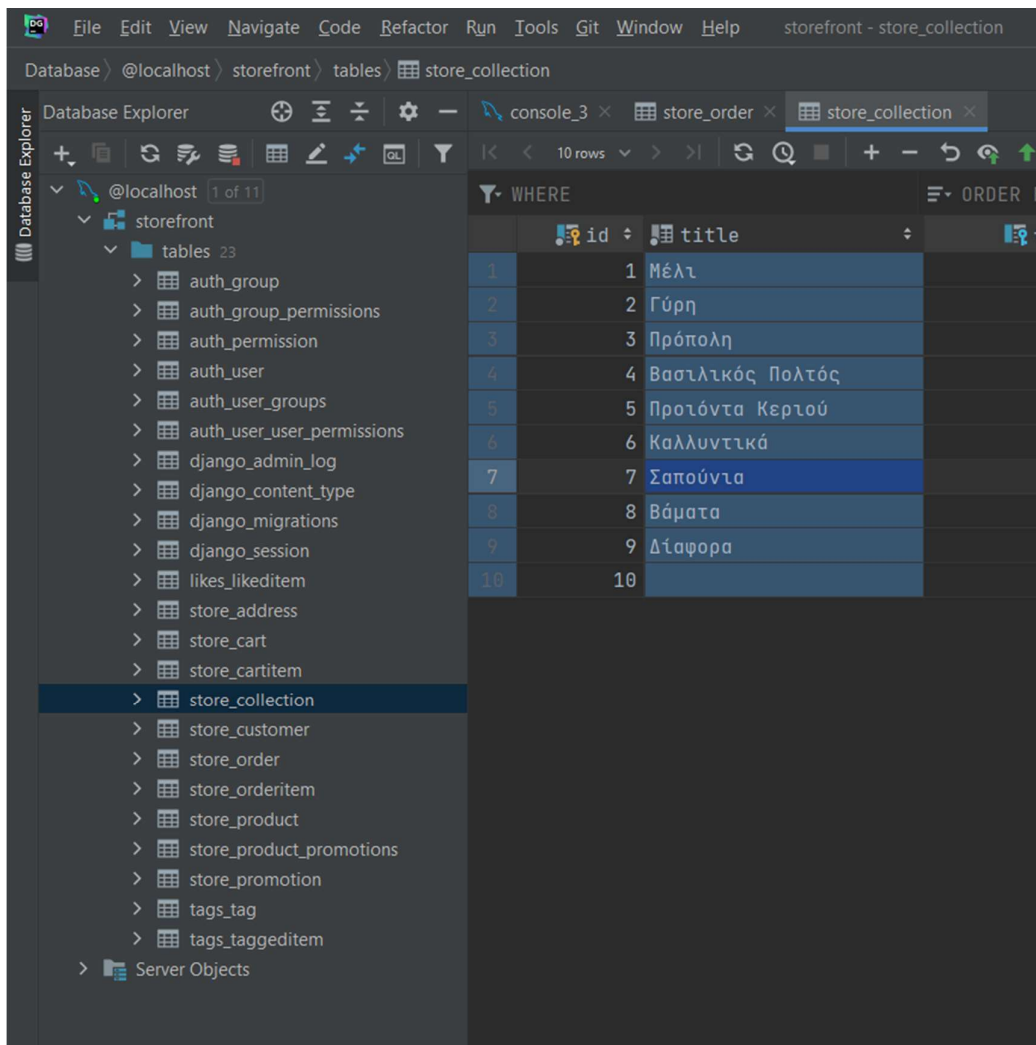
- 3) **product**: Εξωτερικό κλειδί (ForeignKey) που συνδέει το αντικείμενο με ένα προϊόν (Product).
- 4) **quantity**: Ακέραιο πεδίο (PositiveSmallIntegerField) που αντιπροσωπεύει την ποσότητα του προϊόντος που βρίσκεται στο καλάθι.

```
45 MEMBERSHIP_GOLD = 'G'
46
47 MEMBERSHIP_CHOICES = [
48     (MEMBERSHIP_BRONZE, 'Bronze'),
49     (MEMBERSHIP_SILVER, 'Silver'),
50     (MEMBERSHIP_GOLD, 'Gold'),
51 ]
52 first_name = models.CharField(max_length=255)
53 last_name = models.CharField(max_length=255)
54 email = models.EmailField(unique=True)
55 phone = models.CharField(max_length=255)
56 birth_date = models.DateField(null=True, blank=True)
57 membership = models.CharField(
58     max_length=1, choices=MEMBERSHIP_CHOICES, default=MEMBERSHIP_BRONZE)
59
60 def __str__(self):
61     return f'{self.first_name} {self.last_name}'
62
63 class Meta:
64     ordering = ['first_name', 'last_name']
65
66
67 class Order(models.Model):
68     PAYMENT_STATUS_PENDING = 'P'
69     PAYMENT_STATUS_COMPLETE = 'C'
70     PAYMENT_STATUS_FAILED = 'F'
71     PAYMENT_STATUS_CHOICES = [
72         (PAYMENT_STATUS_PENDING, 'Pending'),
73         (PAYMENT_STATUS_COMPLETE, 'Complete'),
74         (PAYMENT_STATUS_FAILED, 'Failed')
75     ]
76
77     placed_at = models.DateTimeField(auto_now_add=True)
78     payment_status = models.CharField(
79         max_length=1, choices=PAYMENT_STATUS_CHOICES, default=PAYMENT_STATUS_PENDING)
80     customer = models.ForeignKey(Customer, on_delete=models.PROTECT)
81
82
83 class OrderItem(models.Model):
84     order = models.ForeignKey(Order, on_delete=models.PROTECT)
85     product = models.ForeignKey(Product, on_delete=models.PROTECT)
86     quantity = models.PositiveSmallIntegerField()
87     unit_price = models.DecimalField(max_digits=6, decimal_places=2)
88
89
90 class Address(models.Model):
91     street = models.CharField(max_length=255)
```

Στιγμιότυπο από store>models.py

## 7.Εισαγωγή δεδομένων στους πίνακες

Για την εισαγωγή δεδομένων χρησιμοποιήσαμε την εφαρμογή DataGrip (μας προτείναν την δωρεάν δοκιμή των 30 ημερών) και εισήγαμε μία εικονική βάση δεδομένων. Τα εικονικά αυτά δεδομένα τα δημιουργήσαμε με το Mockaroo.com όπου προσαρμόσαμε τα δεδομένα βάση των απαιτήσεων μας. Χρησιμοποιήσαμε και τη διαχειριστική διεπαφή Django (Django Admin) όπου εισαγάγαμε δεδομένα απευθείας και από εκεί. Συναντήσαμε πρόβλημα στην εγκατάσταση της εντολής `pip install mysqlclient` το οποίο μας πήρε πολύ καιρό να το λύσουμε αλλά το καταφέραμε μέσω πολλών και διαφορετικών προτάσεων από το youtube.



	id	title
1	1	Μέλι
2	2	Γύρη
3	3	Πρόπολη
4	4	Βασιλικός Πολτός
5	5	Προϊόντα Κεριού
6	6	Καλλυντικά
7	7	Σαπούνια
8	8	Βάμματα
9	9	Διάφορα
10	10	

Στιγμιότυπο από DataGrip

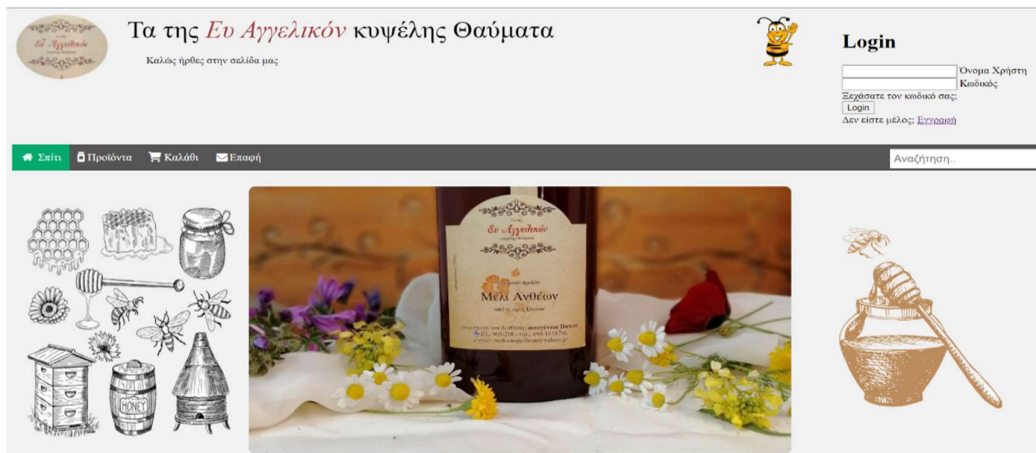
## 8. Δημιουργία templates

Το Ευ Αγγελικόν είναι η βασική σελίδα και ο χρήστης μπορεί να πάει μέσω αυτής στις άλλες. Η εγγραφή αντιστοιχεί στην δημιουργία προφίλ.

Τα Products αντιστοιχούν στην προβολή προϊόντων.

Το Contact στην επικοινωνία με τον Admin.

Το checkout με την ολοκλήρωση της παραγγελίας



```
playground > urls.py > ...
1  from django.urls import path
2  from . import views
3  from django.urls import path
4  |
5  urlpatterns = [
6      path('checkout/', views.checkout, name='checkout'),
7      path('contact/', views.contact, name='contact'),
8      path('home/', views.home, name='home'),
9      path('products/', views.products, name='products'),
10     path('register/', views.register, name='register'),
```



## Εγγραφή

Παρακαλώ συμπληρώστε την παρακάτω φόρμα για να κάνετε εγγραφή

### Email

Συμπληρώστε το email σας

### Κωδικός

Γράψτε κωδικό

### Επανάληψη Κωδικού

Επανάλάβετε τον κωδικό

☒ Remember me

Δημιουργώντας προφίλ, συμφωνείτε με τους παρακάτω [Όρους και Προϋποθέσεις](#).

Ακύρωση

Εγγραφή



### Στοιχεία Παραλήπτη

#### ▲ Ονοματεπώνυμο

Γιάννης Μ. Παπαδόπουλος

#### ✉ Email

john@example.com

#### 📄 Διεύθυνση

Οδός 100

#### 🏠 Πόλη

Αθήνα

#### Γ.Δ.

Αττική

#### Τ.Κ.

10001



## Επικοινωνήστε μαζί μας

### Όνομα

Το όνομα σου.

### Επώνυμο

Το επώνυμο σου

### Θέμα

Πες μας την γνώμη σου...

Υποβολή

**Πληρωμή**

Αποδοχές Κάρτες

Όνομα επων. Κάρτα

Επίσημο Παροδότης

Αριθμός Κάρτας

1111-2222-3333-4444

Μήνας Λήξης

Στοιχείο

Έτος Λήξης

2024

CVV

302

☒ Διαθέσιμη Παράδοση για με Χρέωση

Ολοκλήρωση Παραγγελίας

**Κατάθεση**

Παράβολο 1	913
Παράβολο 2	93
Παράβολο 3	98
Παράβολο 4	92
Σύνολο	996

## 9.Υλοποίηση με το Django framework

Θα αναφέρουμε περιγραφικά την σειρά των εντολών που χρησιμοποιήθηκαν αλλά και που χρειάζονται για να τρέξει ο κώδικας. Από τη στιγμή που θα αποσυμπιεστεί το αρχείο και θα ανοιχτεί με Visual Studio Code θα πρέπει για να είναι λειτουργικό να γίνουν οι εξής ενέργειες.

1. Να δημιουργηθεί το κατάλληλο περιβάλλον .Εντολή στο terminal : `pipenv install`.
2. Να εγκατασταθεί το Debug. Εντολή στο terminal : `pip install debug_toolbar`.
3. Να εγκατασταθεί το MySQLclient. Εντολή στο terminal : `pip install mysqlclient`.
4. Να τρέξουμε την εντολή `python manage.py runserver`

Από την στιγμή όπου δεν υπάρχει κάποιο error θα μας επιστρέψει στο terminal το αντίστοιχο link. Ακολουθώντας το link θα βρεθούμε στον φυλλομετρητή όπου τρέχει στο localhost.

Θα θέλαμε να αναφέρουμε πως χρειάστηκε αρκετός χρόνος για την κατανόηση παραπάνω απαιτήσεων. Η απαιτήσεις έχουν όλες πραγματοποιηθεί δεν είναι όμως στον βαθμό των οποίο εμείς επιθυμούσαμε .Τα δεδομένα τα οποία έχουν εισαχθεί είναι εικονικά και δεν πληρούν τις προδιαγραφές όπου είχαν τεθεί εξαρχής.

## 10. Αρμοδιότητες μελών της ομάδας

Οι αρμοδιότητες είχαν διαχωριστεί ως εξής ο Αντώνης Βερβαινιώτης εργάστηκε ως Front-end developer ενώ η Χανιωτάκη Ευαγγελία σαν back-end developer με αυτόν τον τρόπο καταφέραμε την πραγματοποίηση των παραπάνω αναγκών σε παράλληλο χρόνο.

**Θέμα και πηγές συλλογής απαιτήσεων** Αντώνης Βερβαινιώτης και Χανιωτάκη Ευαγγελία .

**Σχεδιασμός υψηλού επιπέδου** Αντώνης Βερβαινιώτης σχεδισμός use case ,Χανιωτάκη Ευαγγελία ανάλυση.

**Αρχιτεκτονικός Σχεδιασμός** Χανιωτάκη Ευαγγελία και Αντώνης Βερβαινιώτης

**Οντότητες, συσχετίσεις, παραδοχές και ER** Αντώνης Βερβαινιώτης σχεδιασμος και Χανιωτάκη Ευαγγελία ανάλυση .

**Δημιουργία views:** Χανιωτάκη Ευαγγελία .

**Δημιουργία πινάκων :** Χανιωτάκη Ευαγγελία .

**Εισαγωγή δεδομένων στους πίνακες :** Χανιωτάκη Ευαγγελία .

**Δημιουργία templates:** Αντώνης Βερβαινιώτης σχεδιασμος HTML , CSS και Χανιωτάκη Ευαγγελία .

**Υλοποίηση με το Django framework :** Χανιωτάκη Ευαγγελία .

Ευχαριστούμε πάρα πολύ καικαλό καλοκαίρι!